

---

# AMR Application Development with the SAMRAI Library

---

**Rich Hornung**

*Center for Applied Scientific Computing  
Lawrence Livermore National Laboratory*

[hornung@llnl.gov](mailto:hornung@llnl.gov)

[www.llnl.gov/CASC/SAMRAI](http://www.llnl.gov/CASC/SAMRAI)

*Workshop on Adaptive Mesh Refinement  
LACSI Symposium 2004  
Santa Fe, NM  
October 12, 2004*



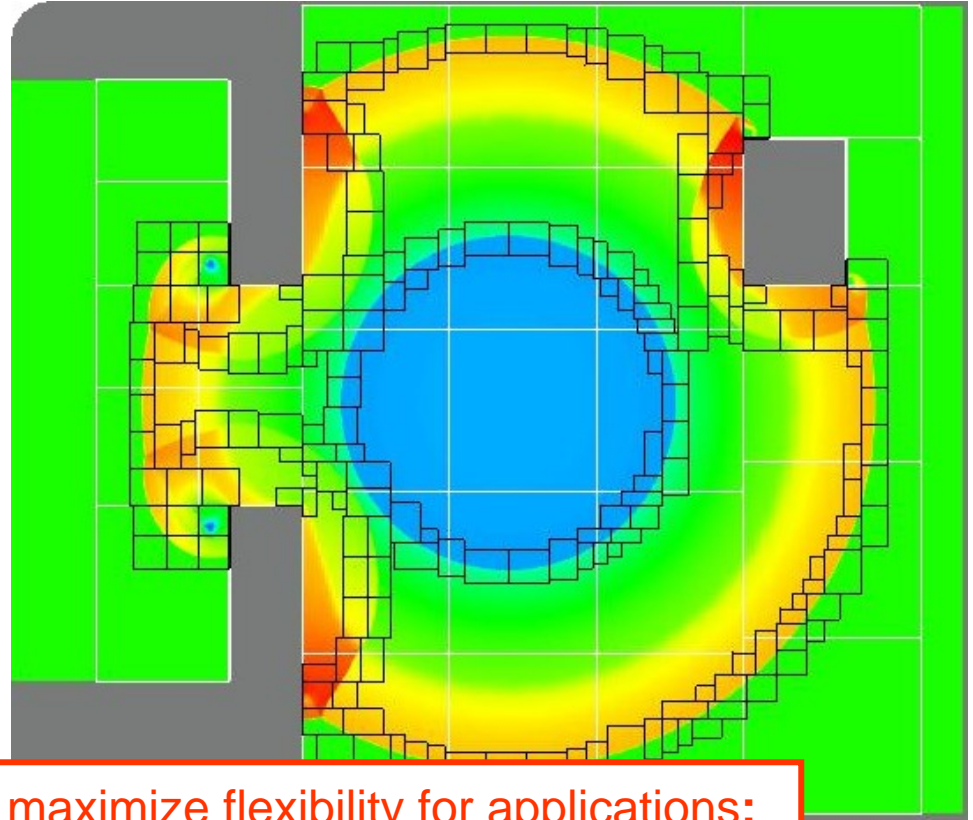
This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.



# Structured AMR employs a dynamic “patch hierarchy”

SAMR mesh and data:

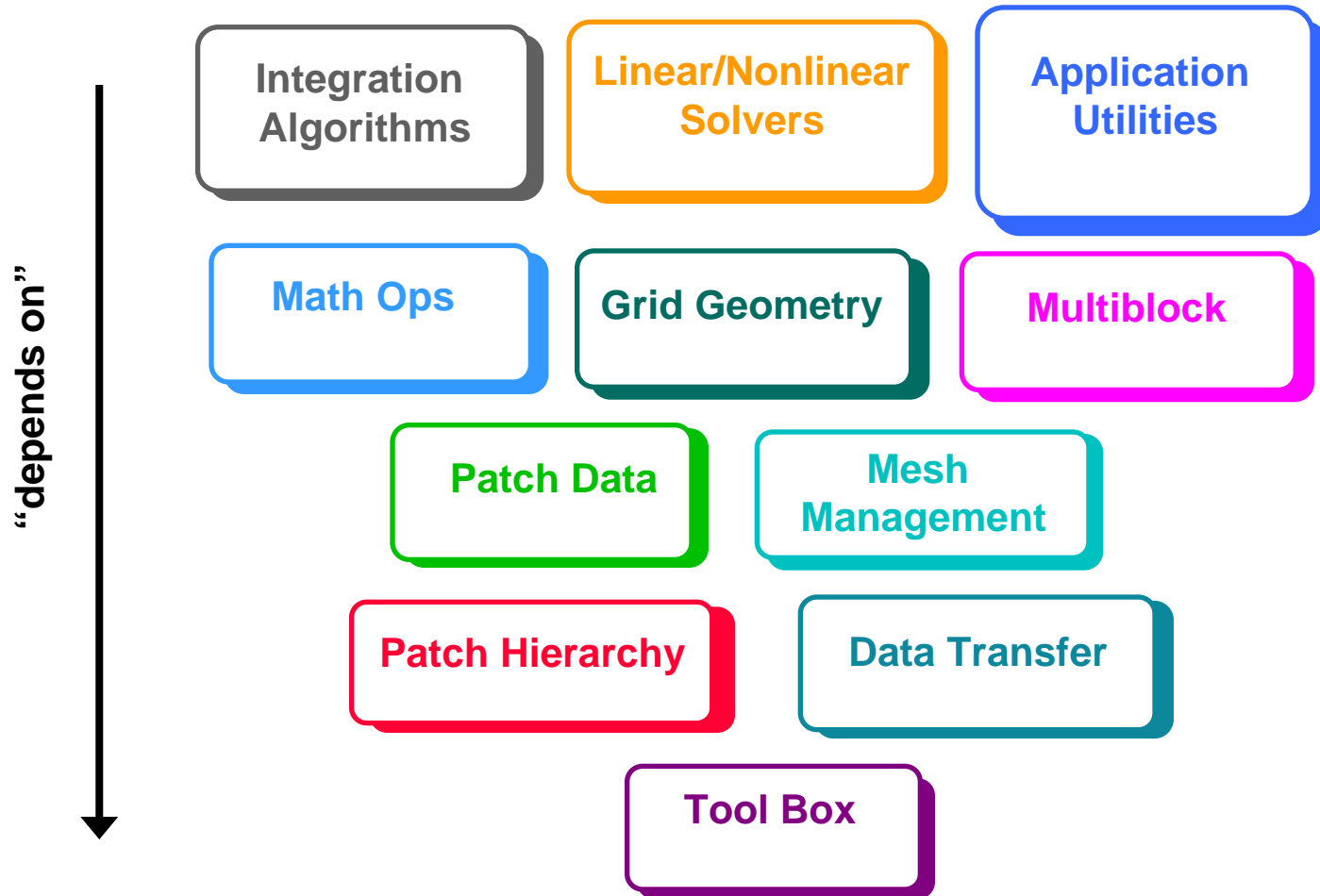
- hierarchy of nested “patch” levels → low overhead mesh description
- cells clustered into patches of varying size → favorable comp/comm volume ratios
- data mapped to patches → simple model of data locality
- any grid system that maps to logically-rectangular index space



SAMRAI goal – minimize constraints & maximize flexibility for applications:

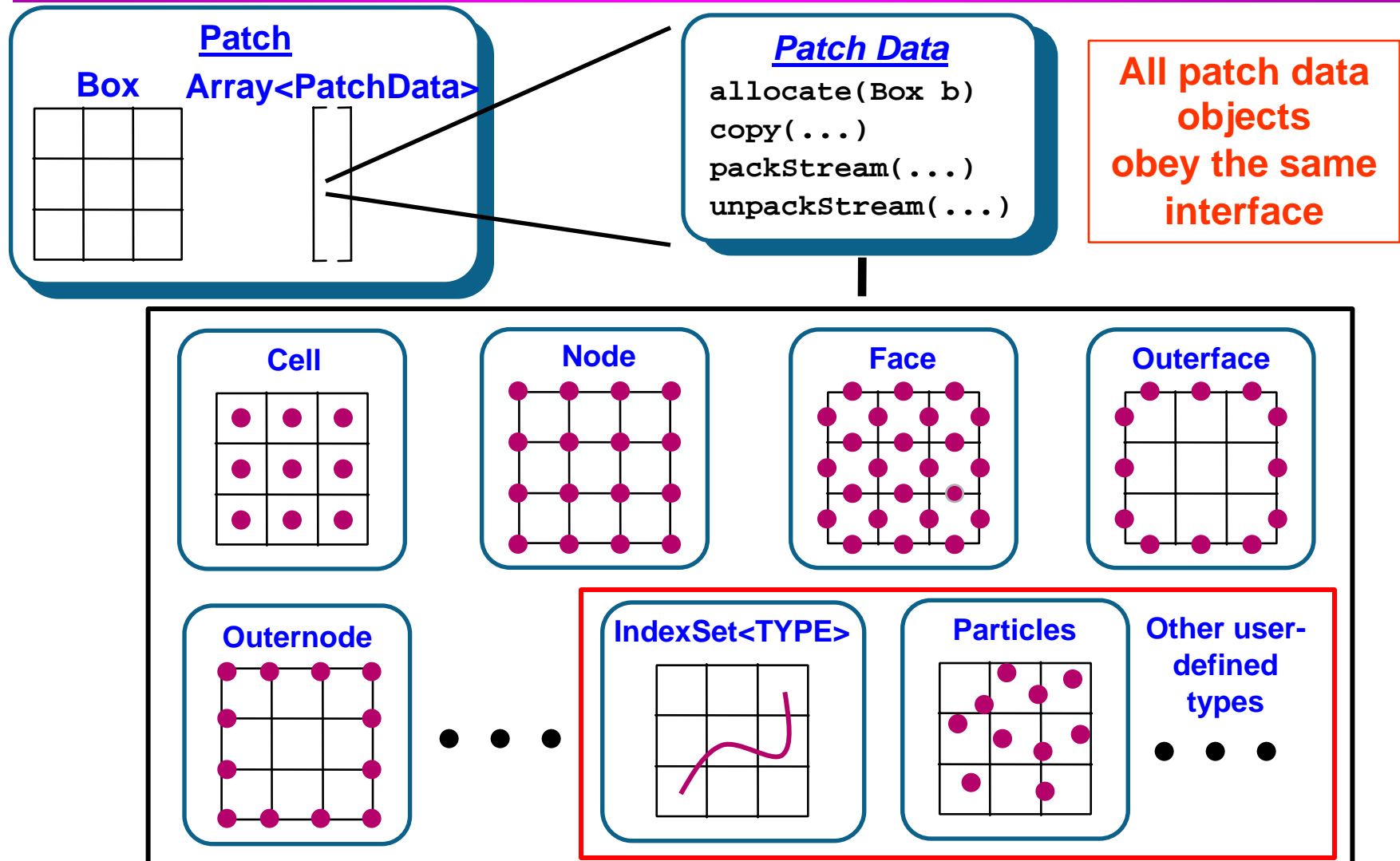
- problem formulation for locally-refined mesh
- (serial) numerical routines for individual patches
- inter-patch data transfer operations (copy, coarsen, refine, time int, ...)

# SAMRAI is an object-oriented “toolbox” of classes for SAMR application development



Hornung, Kohn, “Managing Application Complexity in the SAMRAI Object-oriented Framework”, *Concurrency Computat.: Pract. Exper.* **14**:347-368 (2002)

# A SAMRAI "patch" contains all data on a box region of the computational mesh



# SAMRAI *Variable* and *PatchData* delineate “static” and “dynamic” data concepts

---

Solution algorithms and variables tend to be static

- **Variable object**

- defines a data quantity; type, (centering), (depth)
- abstract base class (interface) attributes:
  - name (string)
  - unique instance id (int)
- creates data object instances (*abstract factory*)
- *Variable objects usually persist throughout computation*

Mesh and data objects tend to be dynamic

- **PatchData object**

- represents data on a “box”
- abstract base class (interface) attributes:
  - interior box (Box)
  - exterior box (Box)
  - ghost cell width (IntVector)
- interface for all data communication (*strategy*)
- (usually) created by factory associated with variable
- *PatchData objects are created and destroyed as mesh changes*

# SAMRAI communication framework centers around three abstractions

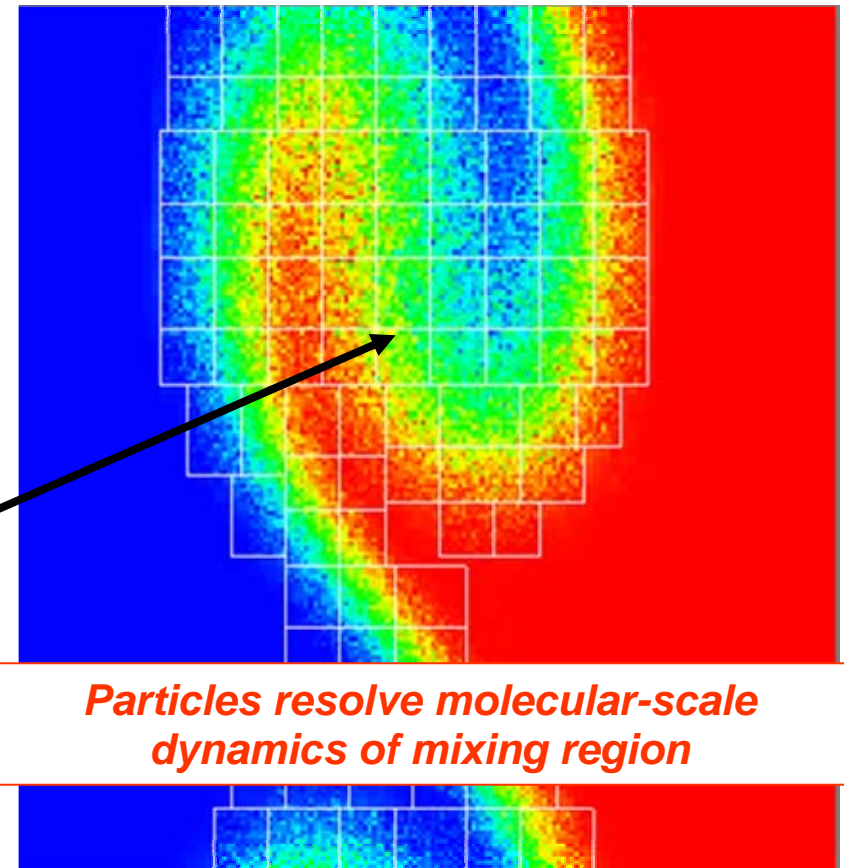
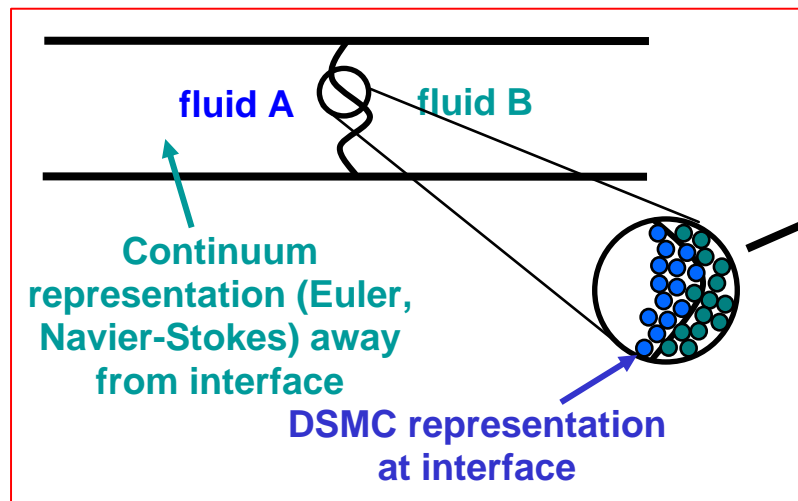
---

- **Communication Algorithm** supports solution-algorithm level description of data transfer phases of computation
  - expressed using variables, coarsen/refine operators, etc.
  - independent of AMR mesh configuration
- **Communication Schedule** manages data transfers for algorithm
  - automatically treats complexity of different data types (e.g., centerings)
  - depends on AMR mesh configuration
- **“Patch Strategy”** is interface to user-defined coarsen/refine operations and boundary conditions

SAMRAI supports data movement involving arbitrary combinations of variable quantities and operations within a single data transfer

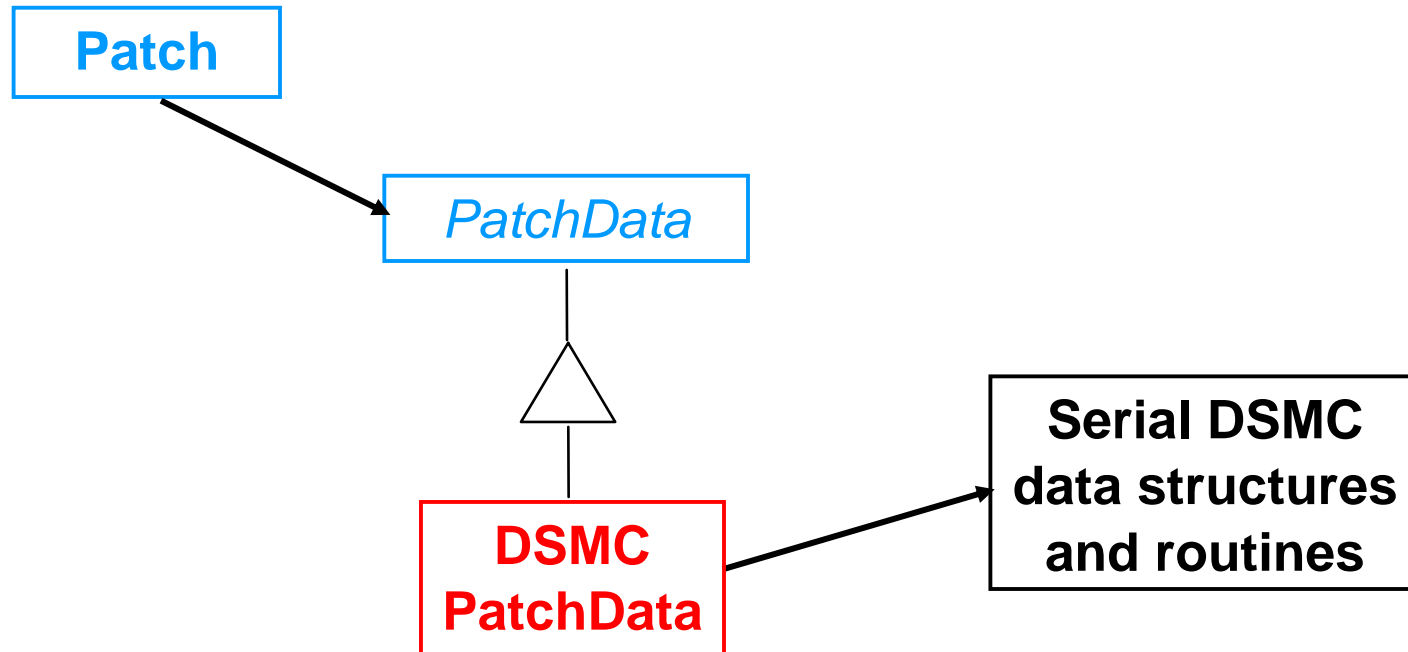
# Adaptive Mesh and Algorithm Refinement (AMAR) refines mesh and numerical model

- AMR is used to refine continuum calculation and focus particles
- Algorithm switches to discrete atomistic method to include physics absent in continuum model



Wijesinghe, Hornung, Garcia, Hadjiconstantinou, “Three-dimensional Hybrid Continuum-Atomistic Simulations for Multiscale Hydrodynamics”, *J. Fluid. Eng* (to appear).

# Pre-existing particle data structures coupled to SAMRAI via patch data interface



```
DsmcPatchData* particles = patch->getPatchData(. . .);  
particles->advance(dt);
```

# Communication algorithms describe data transfers needed for solution method

For example, integration of particle regions requires both continuum and particle boundary data for each patch

- Create algorithm to fill data

```
RefineAlgorithm fill_alg;
```

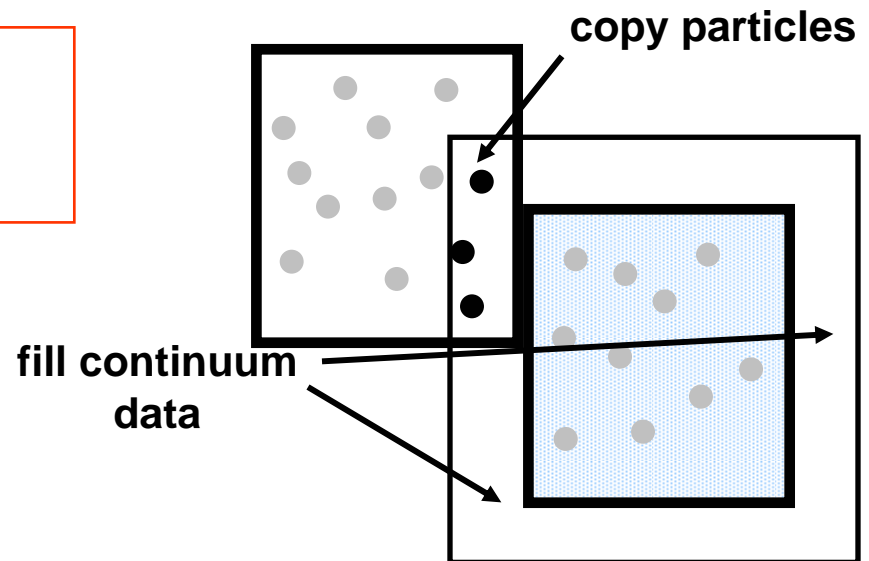
- Register variable operations with algorithm:

- density refined from coarser, copied from fine, BCs set

```
fill_alg.registerRefine(rho_old,           // destination  
                        rho_old, rho_new,   // sources  
                        ..., "CONSERVATIVE_INTERP");
```

- particles copied from neighboring patches

```
fill_alg.registerRefine(particles, // destination  
                        particles,  // source  
                        ...);
```

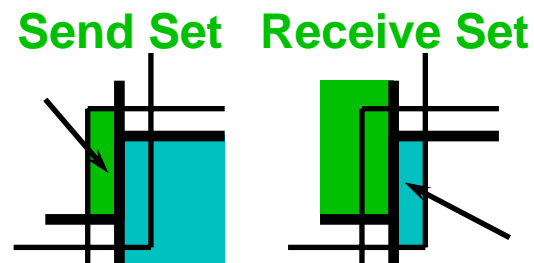
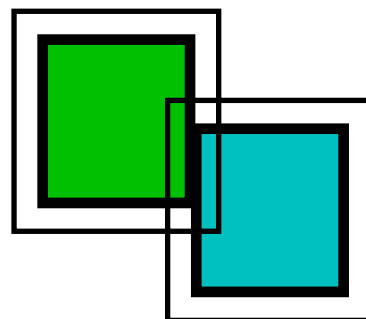


# Communication schedules create and store data dependencies on mesh

- Amortize cost of creating send/receive sets over multiple communication cycles

- Create schedule to fill data

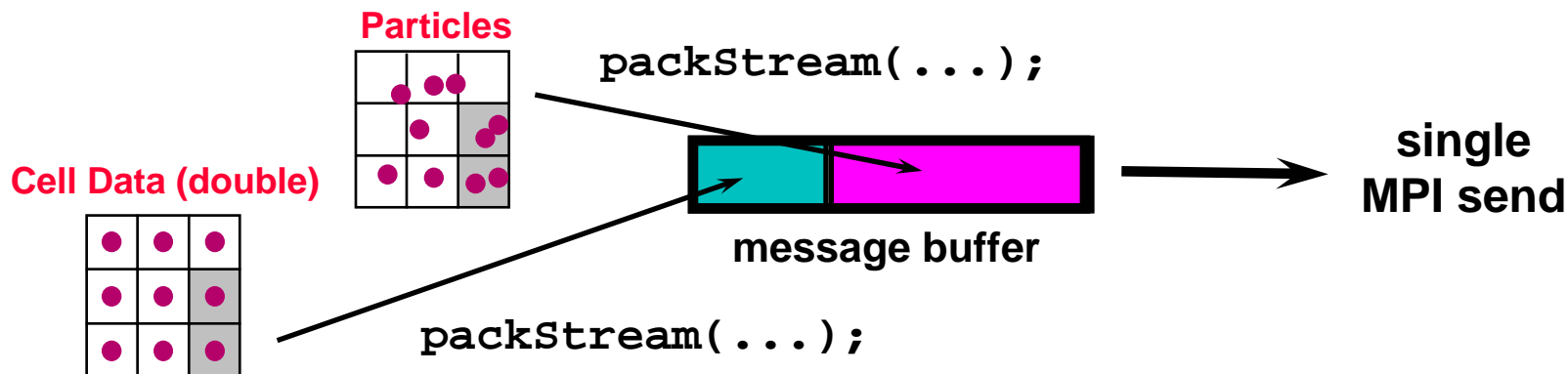
```
RefineSchedule fill_sched =  
    fill_alg.createSchedule(  
        hierarchy, level, ...);
```



- Data from multiple sources is packed into one message stream

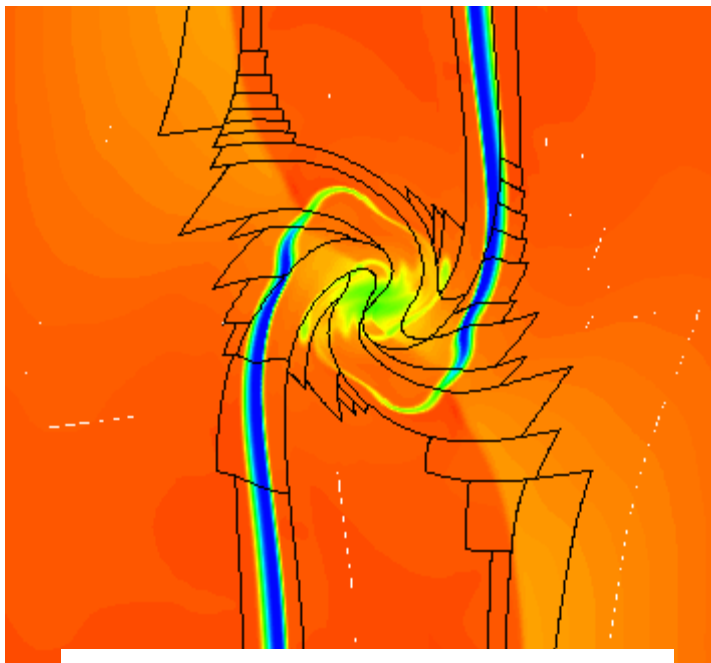
- Invoke data fill operations

```
fill_sched.fillData(time, ...);
```

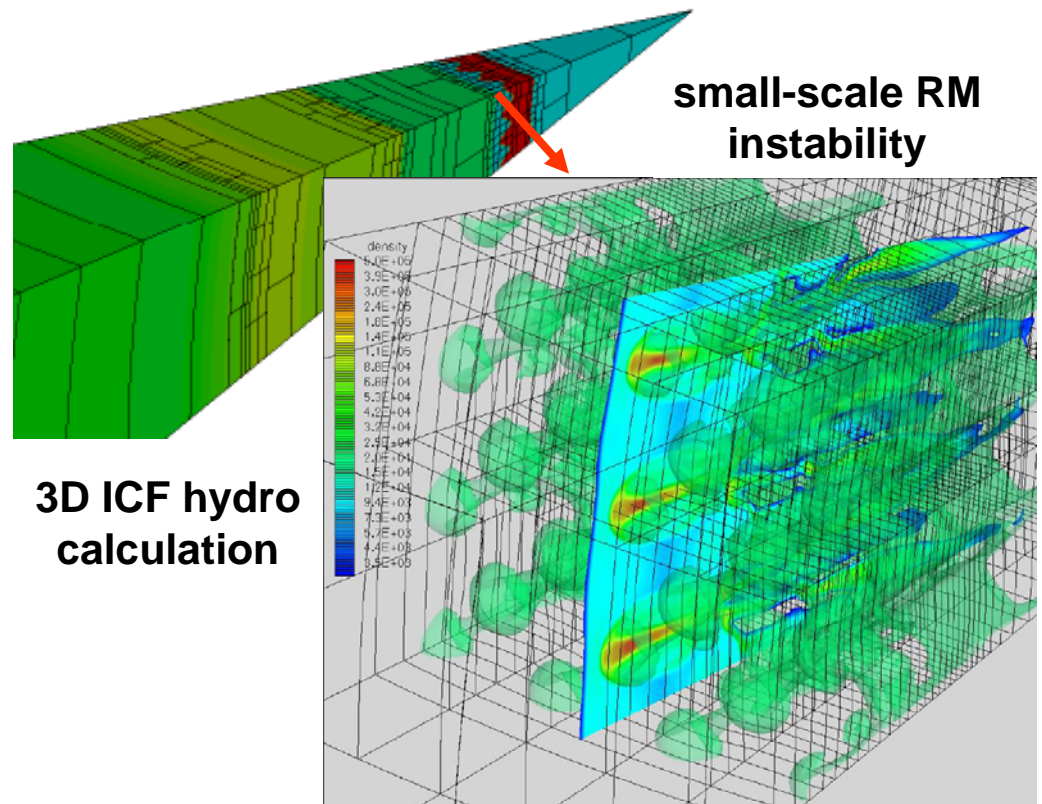


# ALE-AMR combines ALE integration with AMR

- Advantages of ALE (multiple materials, moving interfaces)
- Advantages of AMR (dynamic addition & removal of mesh points)



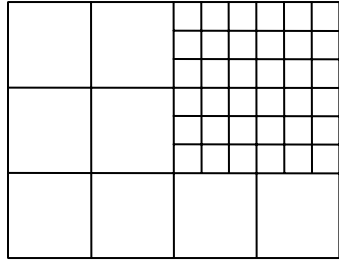
**Moving-deforming AMR grid**



**3D ICF hydro  
calculation**

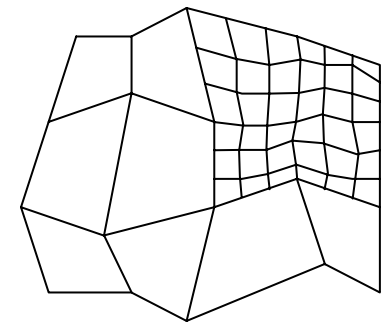
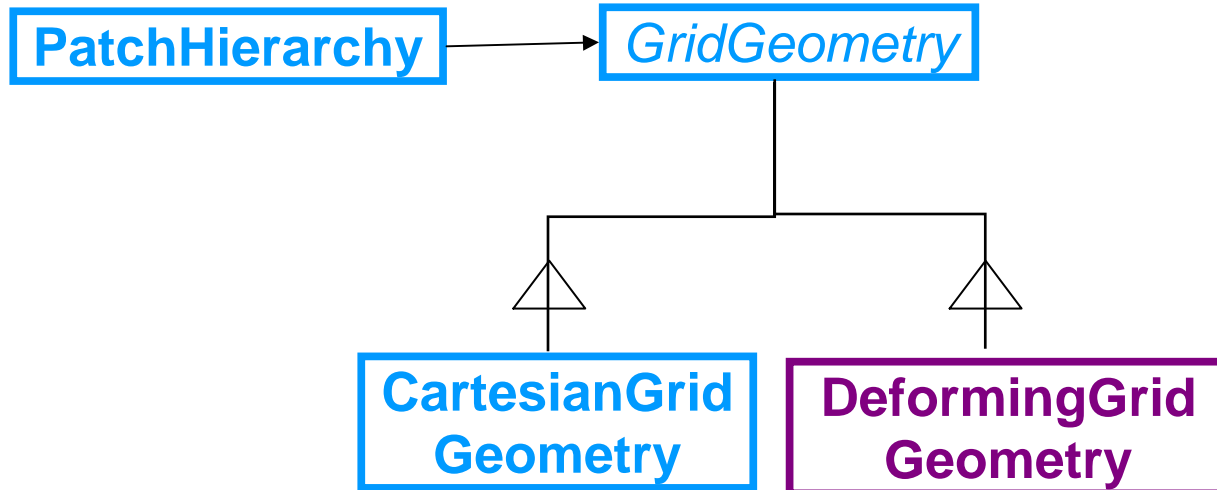
Anderson, Elliott, Pember, "An Arbitrary Lagrangian-Eulerian Methods with Adaptive Mesh Refinement for the Solution of the Euler Equations", *J. Comp. Phys.* **199**(2): 598-617 (2004).

# ALE-AMR manages deforming grids by specializing SAMRAI grid geometry



Manages “index space” coordinates

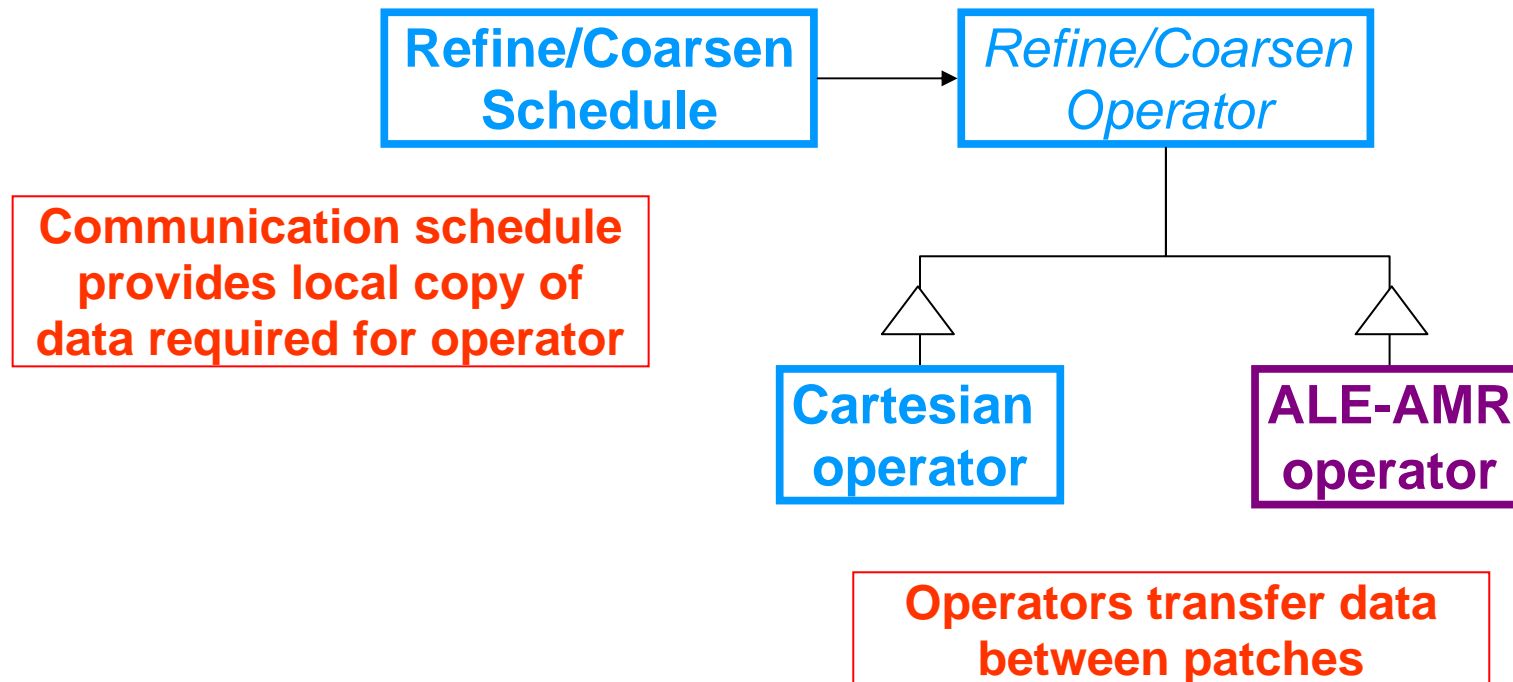
ALE-AMR grid coordinates are presented as “node” patch data



Manages “physical space” coordinates

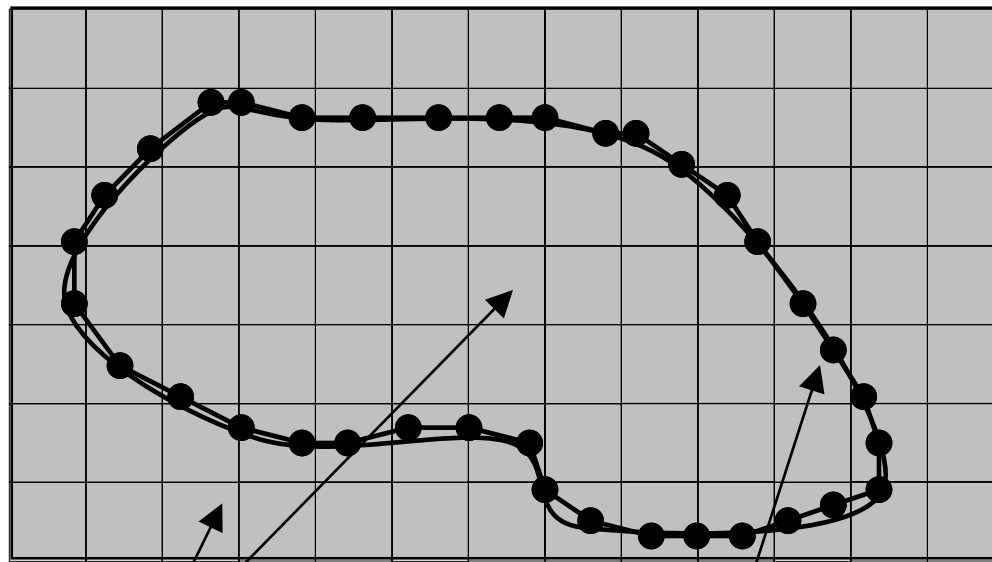
# ALE-AMR specializes interpolation/coarsen operators

- Operator interfaces define interpolation/coarsen operations



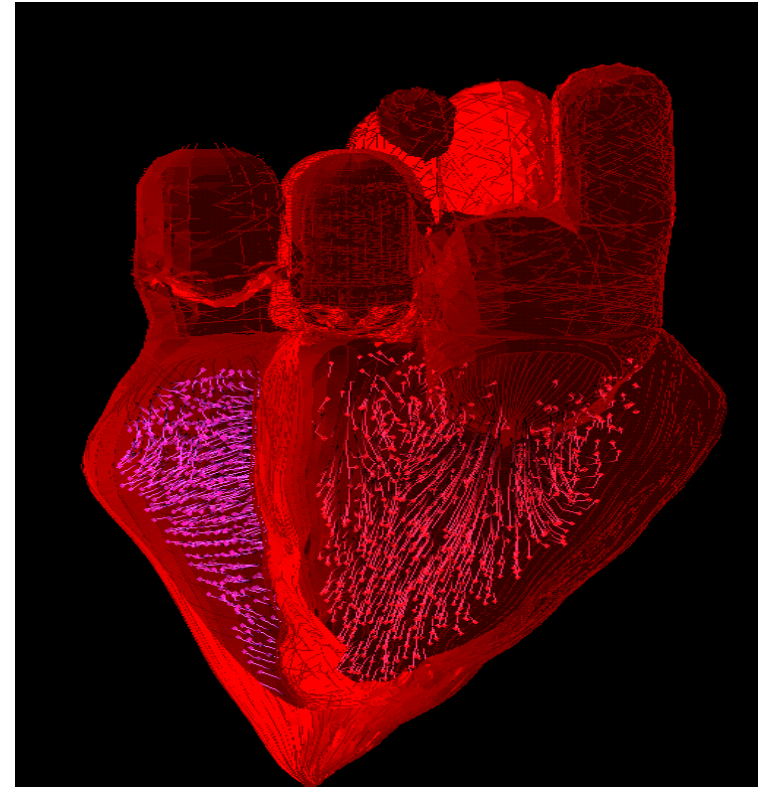
- Interfaces for more complex operations (e.g., those needing multiple patch data quantities) are also used

# Immersed boundary methods model fluid structure interactions



Fluid domain:  
 $\mathbf{u}(\mathbf{x}, t), p(\mathbf{x}, t), \mathbf{f}(\mathbf{x}, t)$

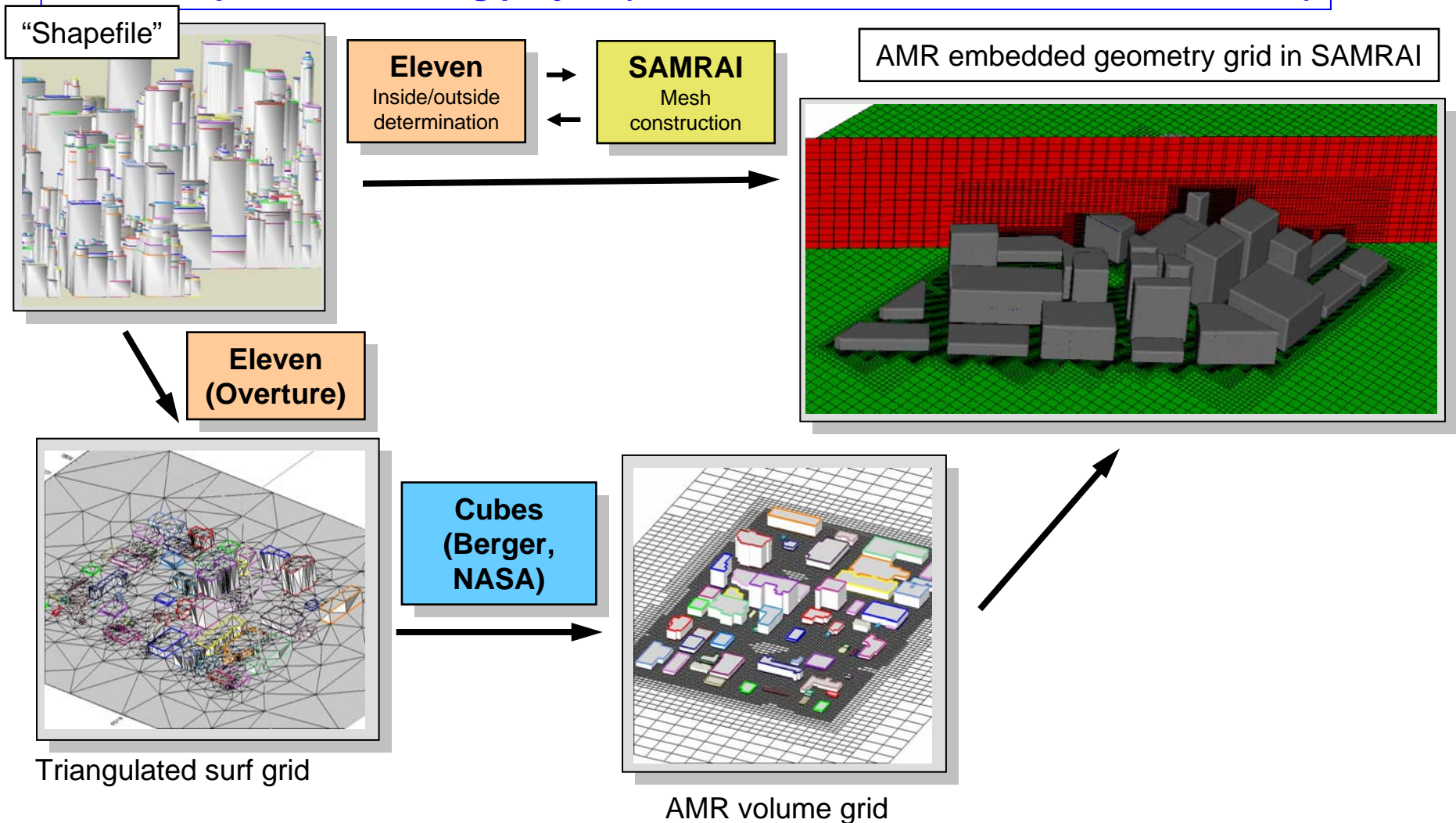
Structure domain:  
 $\mathbf{X}(s, t), \mathbf{F}(s, t)$



**Griffith, Peskin (NYU)** are developing an electrical-mechanical heart model combining immersed boundaries and AMR (SAMRAI)

# SAMRAI supports other embedded geometry using other packages

Urban Dispersion Modeling project (Wissink, Kosovic, Chand, Petersson, et al.)



# SAMRAI index data supports embedded boundary as “patch data”

- *IndexVariable* and *IndexData* classes manage data quantities on irregular index sets

```
IndexVariable<TYPE> ivar("name")
```

```
IndexData<TYPE> idata(Box& box, ghosts)
```

**“TYPE”**

## Required methods

```
TYPE()
```

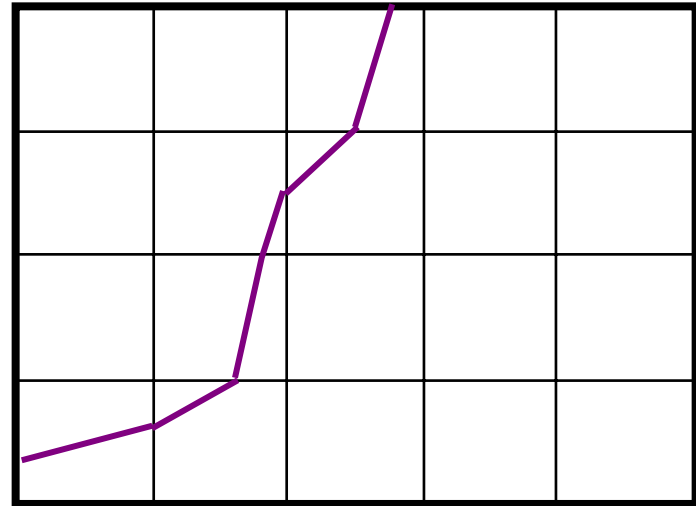
```
TYPE& operator=(const TYPE&)
```

```
getDataStreamSize(Box&)
```

```
packStream(...)
```

```
unpackStream(...)
```

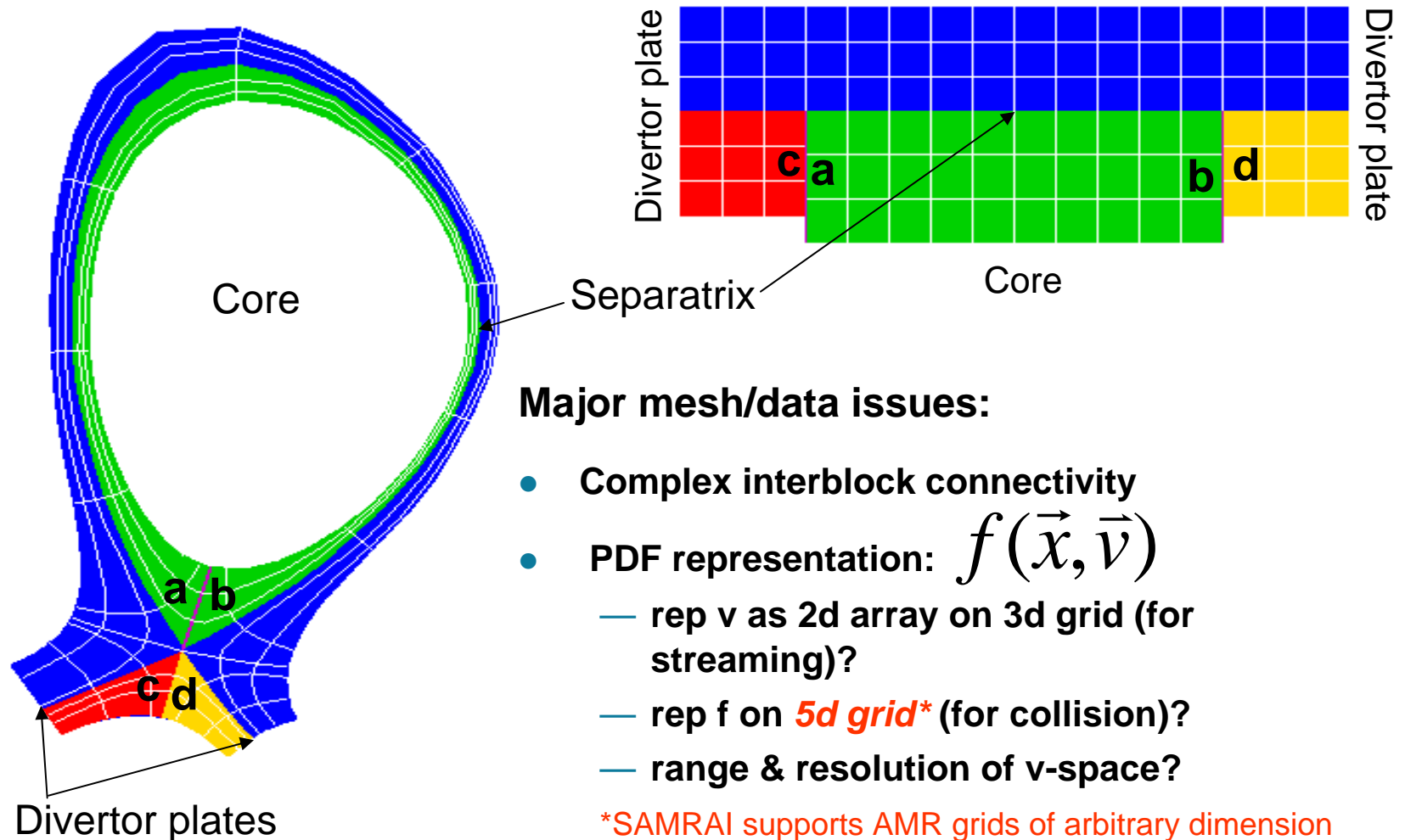
e.g.



**CutCell** type describes internal boundary and state information along boundary

# Gyrokinetic edge plasma code has unique mesh structure requirements

Collaborators: Nevins, Dorr, Hittinger, et al.



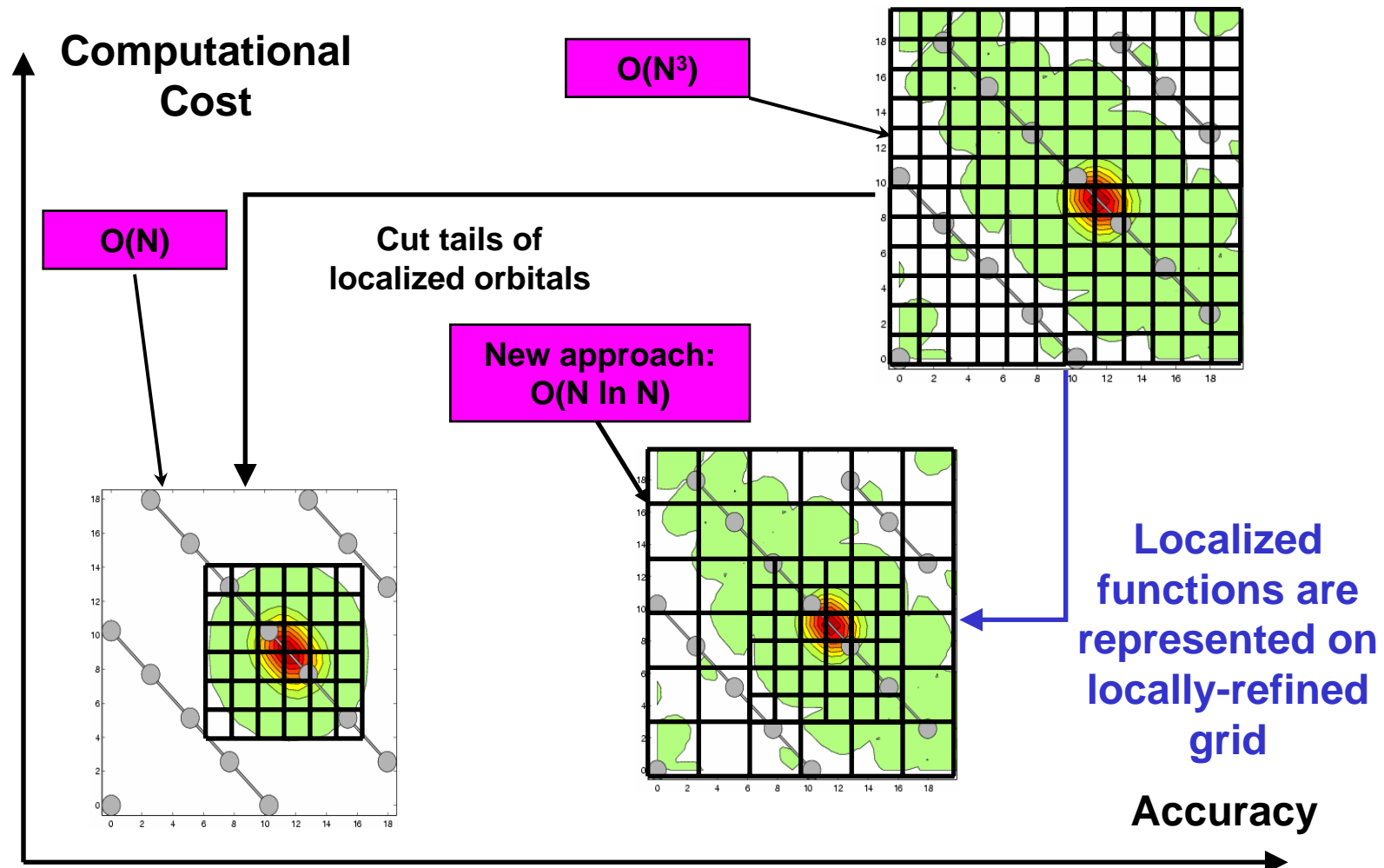
## Major mesh/data issues:

- Complex interblock connectivity
- PDF representation:  $f(\vec{x}, \vec{v})$ 
  - rep  $v$  as 2d array on 3d grid (for streaming)?
  - rep  $f$  on **5d grid\*** (for collision)?
  - range & resolution of  $v$ -space?

\*SAMRAI supports AMR grids of arbitrary dimension

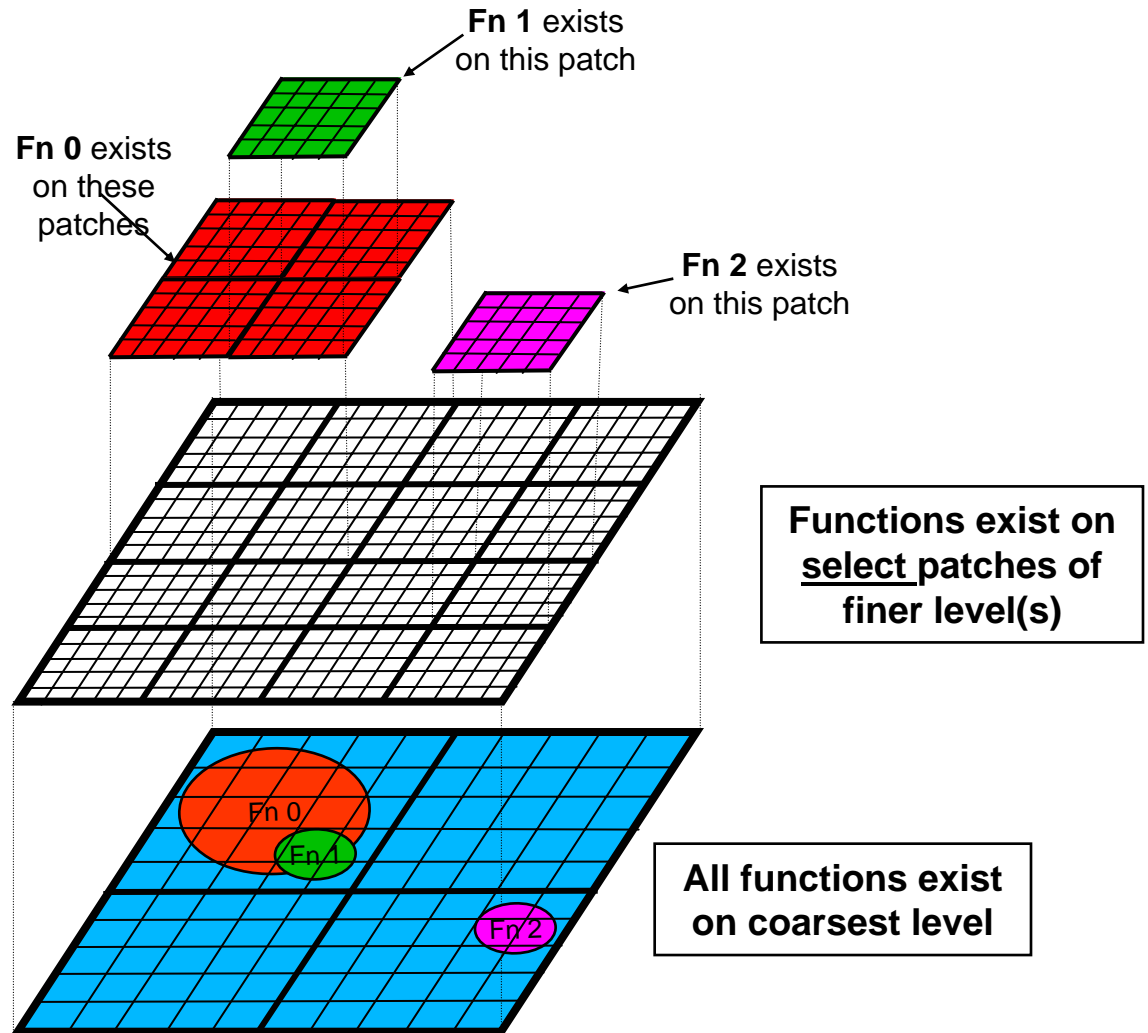
# Using AMR for electronic structures may greatly improve scaling while preserving accuracy

Fattebert, Gygi, Hornung, Wissink



# “Locally-active” data support in SAMRAI

- Typical AMR applications (e.g., CFD) use data quantities defined on all patches in AMR grid hierarchy
- New AMR electronic structures algorithms require data for each function on subset of AMR hierarchy patches
- SAMRAI communication infrastructure extended to support this

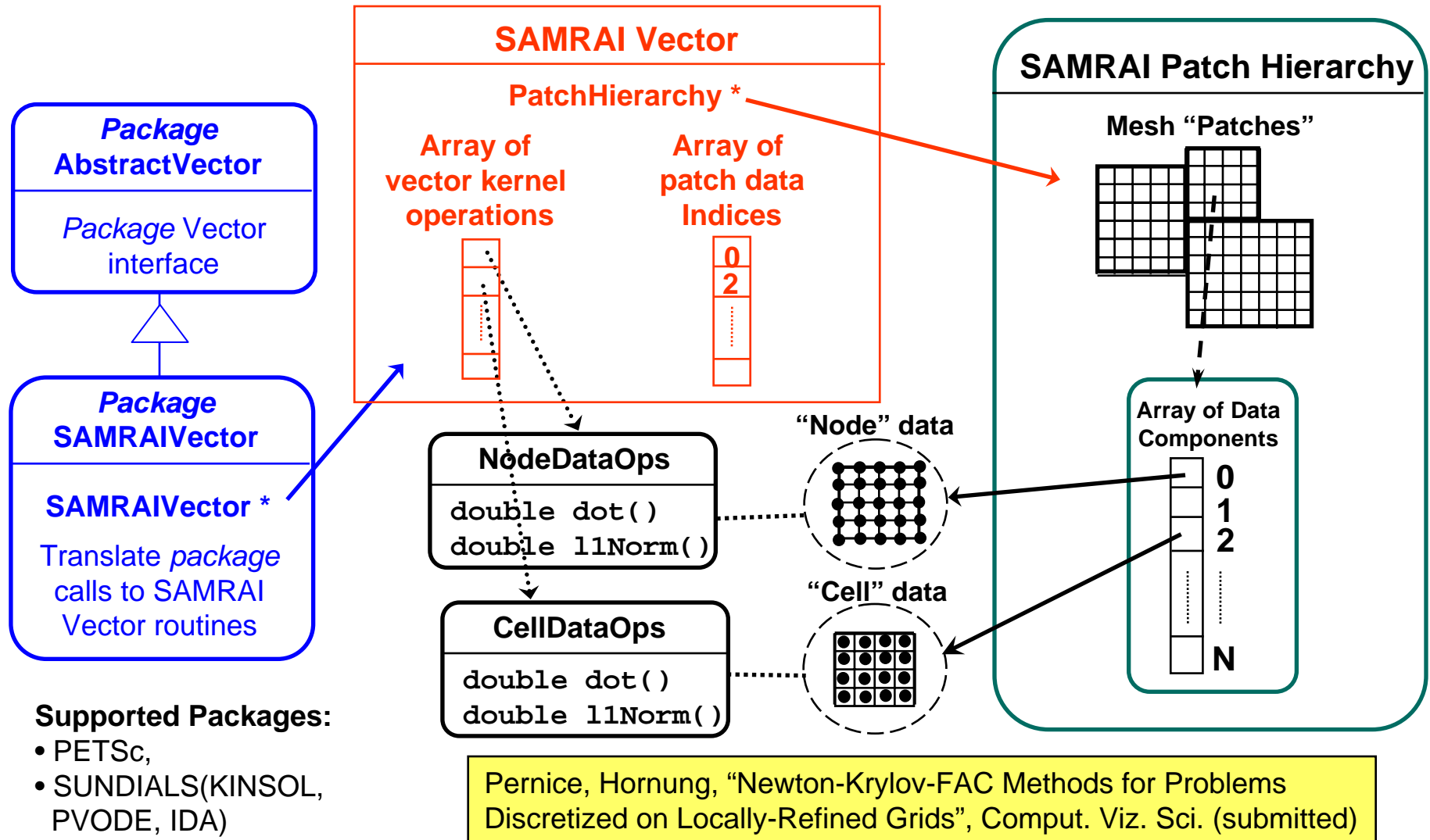


# AMR solver issues

---

- Solver libraries are not typically “AMR-aware”
  - Vector structures don’t support complexity of AMR data configurations
  - AMR requires frequent re-configuration of solvers and vectors
    - vector space dimension changes size as mesh changes
    - only parts of vectors require change during adaptive meshing
- Ideally, we would like to access data “naturally”:
  - nonlinear solver → vector ops, mat-vec ops; mat-vec can be “matrix-free”
  - linear solvers → vector kernels, mat-vec ops; special optimizations
  - applications → discretizations, residual comp; need to know AMR grid
- Loose coupling between vector concept and data storage
  - variable/vector definition independent of mesh is key:
    - for treating frequent changes to vectors during adaptive meshing
    - for providing “natural” data access in various parts of code
  - multiple variable quantities (e.g., different “centerings”) grouped into a single variable is a very useful capability for complex problems

# SAMRAI vectors allow AMR data to be used “natively” by solver libraries



# Concluding remarks

---

- AMR is an increasingly important technology for large-scale science & engineering problems that exhibit behavior requiring fine local grids to resolve
- New applications require expansion of current AMR methodologies
  - New numerical and computational algorithms
  - Combining traditional array-based data with irregular data representations
  - Complex geometry is increasingly important
  - Increased demands on AMR support software libraries
    - **AMR library usage across diverse applications requires decoupling core software infrastructure from specific application needs**
    - **AMR libraries must interoperate with other software packages – solver libraries, grid & geometry generation libraries**
- Many challenges remain:
  - Error estimation, dynamic load balancing, managing AMR overhead
  - Multiphysics problems that combine algorithms with different numerical properties and performance characteristics